REMARKS

Claims 1 and 26 - 27 have been amended. Claims 28 - 29 have been added. No new matter has been introduced with these amendments or added claims, all of which are supported in the specification as originally filed. Claims 1 - 7, 16, 20 - 21, and 26 - 29 are now in the application.

I. Rejection under 35 U.S.C. §103

Page 3 of the Office Action dated December 23, 2005 (hereinafter, "the Office Action") states that Claims 1 - 2, 4 - 5, 16, 20 - 21, and 26 - 27 are rejected under 35 U.S.C. §103(a) as being unpatentable over U. S. Patent 6,792,466 to Saulpaugh et al. (hereinafter, "Saulpaugh"). Page 7 of the Office Action states that Claims 3, 6, and 7 are rejected under 35 U.S.C. §103(a) as being unpatentable over Saulpaugh in view of Extensible Markup Language (XML) 1.0, a publication of W3C (hereinafter, "the XML specification"). These rejections are respectfully traversed.

Applicant has amended his independent Claims 1, 26, and 27 herein to more clearly specify limitations of his invention. In particular, these claims explicitly specify "generating code for the message syntax definitions in the located structured language specification, according to code-generation guidance specified in a dynamically-selected one of a plurality of language-specific code-generation templates that each specify guidance for generating code in a different programming language, the guidance specified as an image of code to be generated in that programming language and comprising syntax indicating where portions of the message

syntax definitions are to be substituted for portions of the specified image ..." (Claim 1, lines 13 - 23, emphasis added).

Applicant respectfully submits that Saulpaugh has no teaching, nor any suggestion, of these claim limitations. The Office Action cites (on Page 2) col. 24, lines 10 - 20 and lines 37 - 45 as "disclos[ing] the functionality of templates". Applicant respectfully disagrees with this characterization of Saulpaugh. Saulpaugh explicitly states, in a number of places, that the "pieces" used by his gate factory to construct a gate in one embodiment are (1) the XML schema and (2) the URI of the service. See Abstract, lines 11 - 13; col. 7, lines 60 - 62; and col. 20, lines 26 - 29. In another embodiment, an authentication credential is also used. See Abstract, lines 13 - 16; col. 7, lines 62 - 65; and col. 20, lines 29 - 32. However, Applicant's claimed technique uses "message syntax definitions in the ... structured language specification [e.g., a schema]" (Claim 1, lines 13 - 14) and "code generation guidance specified in a ... template" (Claim 1, lines 14 - 15), and Applicant finds no mention of anything used by Saulpaugh's gate factory that corresponds to Applicant's "language-specific code-generation templates".

Furthermore, Applicant has clarified his independent claims to more clearly specify that the template used for code generation is dynamically selected from among "a plurality of language-specific code-generation templates that each specify [code generation] guidance ... in a different programming language" (Claim 1, lines 13 - 16, emphasis added). Saulpaugh has no teaching, nor any suggestion, of a code generation process that can be used in this way to generate code in a dynamically-selected programming language (that is, in a programming

language for which code-generation guidance is specified in a dynamically-selected template that is selected from among a plurality of templates for different programming languages).

Instead, the text from col. 24, lines 10 - 20 that is cited for teaching "template functionality" discusses reusing pieces of the constructed gates. See col. 24, lines 12 - 14, stating that "Certain portions of each gate [i.e., each message endpoint] may be the same, and thus may be reused from gate to gate, such as parts of the message verification code." (emphasis added). Lines 16 - 17 refer to these reused pieces as "common code", and lines 18 - 20 cite an example where the reused pieces are for handling the message layer over varying transports of different gates.

Whereas Saulpaugh's reused pieces are common code resulting from the code generation process, Applicant's templates are not something created during the code generation process, but rather are an input to the code generation process. (See Claim 1, lines 13 - 15, specifying that the code being generated is generated "according to" code-generation guidance in the selected template.)

The text from col. 24, lines 37 - 45 that is cited for teaching "template functionality" refers to a "Java dynamic proxy class". Applicant respectfully submits that this text is not relevant to his claimed invention. This passage from Saulpaugh is discussing a scenario where the code that might be generated would run "in the same Java Virtual Machine as the client application", and explaining that use of reflection "may be faster than sending a message" (col.

24, lines 38 - 43). That is, if the code to be generated will run in the same machine as the code with which it will interact, Saulpaugh states that it is more efficient to simply query the (local) object for its values than to generate code (a "gate") to send a message to that local object. This is not relevant to "language-specific code-generation templates ..." as such are specified in Applicant's independent claims.

On p. 5 of the Office Action, this text from col. 24, lines 37 - 45 is discussed again, where the Office Action states that Saulpaugh "does not explicitly disclose that the 'Java dynamic proxy class' is created with respect to the definitions in the structured language specification", and then states that "It would have been obvious ... to replace the functionality of the 'gate factory' ..." (emphasis added). Applicant respectfully submits that this is clearly distinct from his claimed invention. That is, if it is "obvious" to replace Saulpaugh's "gate factory", i.e., Saulpaugh's code generator functionality, then this result -- which therefore *has no code generator*, and is using reflection instead of code generation -- is something distinct from Applicant's claimed class library [code] generating method/system/computer program product.

Also on p. 5 of the Office Action, the Examiner admits that Saulpaugh does not "explicitly" disclose the creation of a class library. This is the subject matter to which Applicant's claims are directed, and given that Saulpaugh does not disclose this function, Applicant respectfully submits that his independent claims, which each specify "programmatically generating a class library" in their preamble, are patentable over Saulpaugh.

Page 2 of the Office Action states that "use of 'class libraries' was well know[n] in the art". Applicant respectfully submits that his claimed invention is not directed toward use of class libraries, but rather to *programmatically* generating class libraries.

Accordingly, Applicant respectfully submits that his independent Claims 1, 26, and 27 are patentable over Saulpaugh, and that his dependent Claims 2 - 7, 16, and 20 - 21 (as well as added Claims 28 - 29) are patentable (*inter alia*) by virtue of their dependency thereupon. The Examiner is therefore respectfully requested to withdraw the §103 rejections.

II. Conclusion

Applicant respectfully requests reconsideration of the pending rejected claims, withdrawal of all presently outstanding rejections, and allowance of all remaining claims at an early date.

Respectfully submitted,

Marcia L. Doubet

Marcia L. Doubet
Attorney for Applicant
Reg. No. 40,999

Customer Number for Correspondence: 43168
Phone: 407-343-7586
Fax: 407-343-7587